Universidad de San Carlos de Guatemala.

-USAC-

Centro Universitario de Oriente.

-CUNORI-

Facultad de Ingeniería.

Arquitectura de computadores y ensambladores I

Ing. Carlos David Ardón Muñoz

**Conexión Cliente-Servidor con Arduino.**

Luis Felipe Dubón Obando

201442782

Cesar Emilio Casasola Miranda

201244716

Cristian Fernando Segura Martínez

201146144

09 de noviembre de 2016

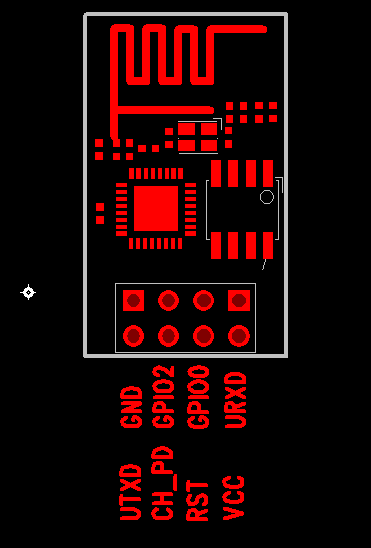
Chiquimula, Guatemala.

**MARCO TEÓRICO**

**MÓDULO WIFI ESP8266EX**

Módulo WiFi con protocolo TCP/IP que puede dar conexión a red inalámbrica a cualquier microcontrolador. El firmware de fábrica trae un juego de comandos AT que permite configurar el módulo, conectarse a la red y enviar datos. Puede configurarse como cliente, servidor y cliente-servidor. Trabaja a 3.3V por lo que es necesario regular el voltaje especialmente para el pin RX.

Cuenta con procesador RISC integrado de 32 bits de baja potencia el cual puede utilizarse como procesador de aplicaciones. A continuación su diagrama y descripción de los pines:



|  |  |  |
| --- | --- | --- |
| Pin | Nombre | Descripción |
| 1 | GND | Tierra |
| 2 | U0TXD | UART0 Transmisión (Tolerante a 5 volts) |
| 3 | GPIO2 | Tiene pull-up interno. |
| 4 | CHIP\_EN | Chip Enable, se activa en alto |
| 5 | GPIO0 | Tiene pull-up interno. |
| 6 | EXT\_RSTB | Señal de Reset externa. Cuando se conecta a corriente, se reinicializa el módulo. |
| 7 | U0RXD | UART0 Recepción (No es tolerante a 5v) |
| 8 | VDD | +3.3V entrada de alimentación |

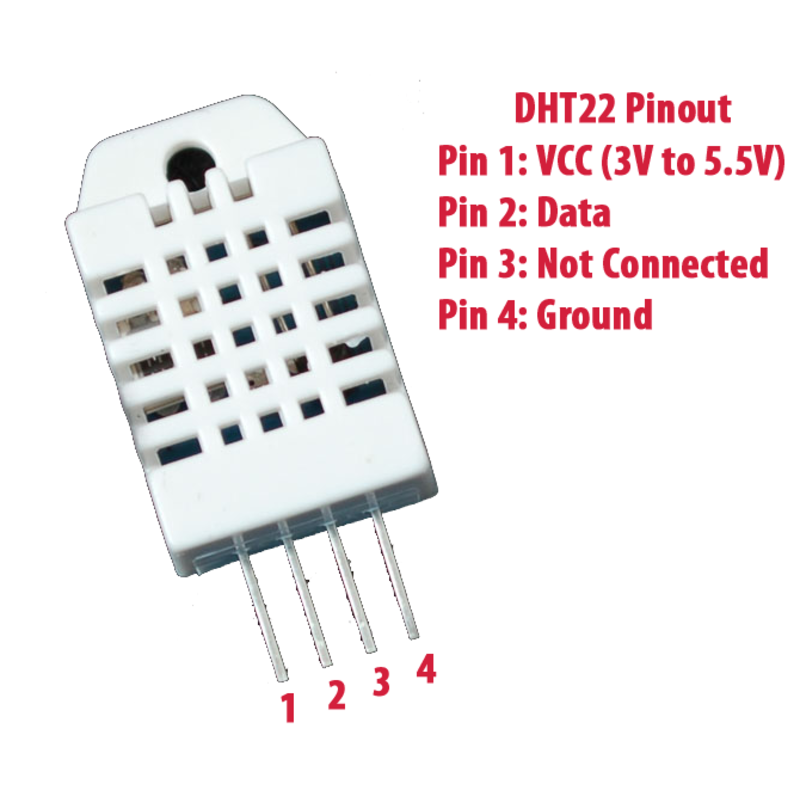
**COMANDOS AT**

Utilizados para configurar y ejecutar funciones en el ESP8266. Entre los más comunes están:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **COMANDO** | | **RESPUESTA** | **FUNCIÓN** | |
| AT | | OK | Prueba si el módulo responde correctamente | |
| AT+CWMODE=modo | | OK | Establece el módulo en el modo dado 1 = Modo estación (cliente) 2 = Modo AP (huésped) 3 = Modo AP + Estación (modo dual) | |
| AT+CWLAP | | AT+CWLAP:ecn,ssid,rssi,mac OK | Lista los Acess Points disponibles para conectarse. ecn: codificación, puede ser: 0 = Abierto 1 = WEP  2 = WPA PSK 3 = WPA2 PSK 4 = WPA WPA2 PSK ssid: String que contiene el SSID del AP rssi: Fuerza de la señal mac: String que contiene la dirección MAC | |
| AT+CWJAP=ssid,pwd | | OK | El módulo se conecta a la red con el nombre ssid indicado y la contraseña pwd suministrada | |
| AT+CIPSTART=type,addr,port | | OK | Empieza una conexión como cliente (en modo conexión única) type: puede ser "TCP" o "UDP" addr: String que contenga la dirección IP remota port: String que contenga el puerto remoto | |
| AT + CIPCLOSE - Cierra la conexión TCP o UDP | | | | |
| AT+CIPSEND=length | SEND OK | | | Establece la longitud de datos a enviarse (máximo 2048). Para un envío normal (modo conexión única) |
| AT+CIPSEND |  | | | Envía datos sin adornos cada 20ms. El módulo retorna ">" después ejecutar el comando, si se recibe el comando "+++" se regresa al modo comando. |

**DHT22**

Módulo sensor de temperatura y humedad relativa de salida digital, también conocido como AM302. Utiliza una exclusiva técnica de recolección de señales digitales y tecnología de lectura de humedad, asegurando su confiabilidad y estabilidad. Sus elementos sensoriales están conectados con un único chip de 8 bits. Su alimentación de energía puede variar entre 3.3-6V. La señal de salida es señal digital vía un único bus. Su elemento sensorial es un capacitor de polímero. Su período de sensibilidad tiene un promedio de 2 segundos.

****

El modelo que se utilizó no tiene el pin 3.

**ARDUINO**

Arduino es una plataforma de electrónica open-source basada en hardware y software fácil de usar. Las placas Arduino son capaces de leer entradas como la luz capturada por un sensor, la pulsación de un botón o un mensaje de Twitter, y encender una salida para, por ejemplo: activar un motor, encender una LED, publicar algo en línea. El lenguaje en el que se programa Arduino es lenguaje de programación Arduino, el cual es basado en Wiring, un framework open-source para programación de microcontroladores. Su IDE está escrito en Java, por lo que es sumamente portable.

Existe un buen número de placas Arduino con características que se adaptan a proyectos, usuarios o necesidades específicas. Para desarrollar este proyecto, utilizamos Arduino Mega 2560.

El Arduino Mega 2560 es una microcontrolador basado en el ATmega2560. Cuenta con 54 pines digitales de entrada / salida (de los cuales 15 se pueden utilizar como salidas PWM), 16 entradas analógicas, 4 UART (puertos serie de hardware), un oscilador de cristal de 16 MHz, una conexión USB, un conector de alimentación, una cabecera ICSP y un botón de reinicio.

# PHP

PHP (acrónimo recursivo de PHP: Hypertext Preprocessor) es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML.

Bien, pero ¿qué significa realmente? Un ejemplo nos aclarará las cosas:

**Ejemplo #1 Un ejemplo introductorio**

<!DOCTYPE HTML>  
<html>  
 <head>  
 <title>Ejemplo</title>  
 </head>  
 <body>  
  
 <?php  
 echo "¡Hola, soy un script de PHP!";  
 ?>  
  
 </body>  
</html>

En lugar de usar muchos comandos para mostrar HTML (como en C o en Perl), las páginas de PHP contienen HTML con código incrustado que hace "algo" (en este caso, mostrar "¡Hola, soy un script de PHP!). El código de PHP está encerrado entre las etiquetas especiales de comienzo y final <?php y ?> que permiten entrar y salir del "modo PHP".

Lo que distingue a PHP de algo del lado del cliente como Javascript es que el código es ejecutado en el servidor, generando HTML y enviándolo al cliente. El cliente recibirá el resultado de ejecutar el script, aunque no se sabrá el código subyacente que era. El servidor web puede ser configurado incluso para que procese todos los ficheros HTML con PHP, por lo que no hay manera de que los usuarios puedan saber qué se tiene debajo de la manga.

Lo mejor de utilizar PHP es su extrema simplicidad para el principiante, pero a su vez ofrece muchas características avanzadas para los programadores profesionales. No sienta miedo de leer la larga lista de características de PHP. En unas pocas horas podrá empezar a escribir sus primeros scripts

# MySQL

**MySQL** es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual GPL/Licencia comercial por Oracle Corporation y está considerada como la base datos open source más popular del mundo1 2 , y una de las más populares en general junto a Oracle y Microsoft SQL Server, sobre todo para entornos de desarrollo web.

MySQL fue inicialmente desarrollado por MySQL AB (empresa fundada por David Axmark, Allan Larsson y Michael Widenius). MySQL A.B. fue adquirida por Sun Microsystems en 2008, y ésta a su vez fue comprada por Oracle Corporation en 2010, la cual ya era dueña desde 2005 de Innobase Oy, empresa finlandesa desarrolladora del motor InnoDB para MySQL.

Al contrario de proyectos como Apache, donde el software es desarrollado por una comunidad pública y los derechos de autor del código están en poder del autor individual, MySQL es patrocinado por una empresa privada, que posee el copyright de la mayor parte del código. Esto es lo que posibilita el esquema de doble licenciamiento anteriormente mencionado. La base de datos se distribuye en varias versiones, una *Community*, distribuida bajo la Licencia pública general de GNU, versión 2, y varias versiones *Enterprise*, para aquellas empresas que quieran incorporarlo en productos privativos. Las versiones *Enterprise* incluyen productos o servicios adicionales tales como herramientas de monitorización y soporte oficial. En 2009 se creó un *fork* denominado MariaDB por algunos desarrolladores (incluido algunos desarrolladores originales de MySQL) descontentos con el modelo de desarrollo y el hecho de que una misma empresa controle a la vez los productos MySQL y Oracle Database.

Está desarrollado en su mayor parte en ANSI C y C++. Tradicionalmente se considera uno de los cuatro componentes de la pila de desarrollo LAMP y WAMP.

MySQL es usado por muchos sitios web grandes y populares, como Wikipedia, Google (aunque no para búsquedas), Facebook, Twitter, Flickr, y YouTube.

# Servidor HTTP Apache

El **servidor HTTP Apache** es un servidor web HTTP de código abierto, para plataformas Unix (BSD, GNU/Linux, etc.), Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.12 y la noción de sitio virtual. Cuando comenzó su desarrollo en 1995 se basó inicialmente en código del popular NCSA HTTPd 1.3, pero más tarde fue reescrito por completo. Su nombre se debe a que alguien quería que tuviese la connotación de algo que es firme y enérgico pero no agresivo, y la tribu Apache fue la última en rendirse al que pronto se convertiría en gobierno de EEUU, y en esos momentos la preocupación de su grupo era que llegasen las empresas y "civilizasen" el paisaje que habían creado los primeros ingenieros de internet. Además Apache consistía solamente en un conjunto de parches a aplicar al servidor de NCSA. En inglés, *a patchy server* (un servidor "parcheado") suena igual que *Apache Server*.

El servidor Apache es desarrollado y mantenido por una comunidad de usuarios bajo la supervisión de la Apache Software Foundation dentro del proyecto HTTP Server (httpd).

Apache presenta entre otras características altamente configurables, bases de datos de autenticación y negociado de contenido, pero fue criticado por la falta de una interfaz gráfica que ayude en su configuración.

Apache tiene amplia aceptación en la red: desde 1996, Apache, es el servidor HTTP más usado. Jugó un papel fundamental en el desarrollo fundamental de la World Wide Web y alcanzó su máxima cuota de mercado en 2005 siendo el servidor empleado en el 70% de los sitios web en el mundo, sin embargo ha sufrido un descenso en su cuota de mercado en los últimos años. (Estadísticas históricas y de uso diario proporcionadas por Netcraft). En 2009 se convirtió en el primer servidor web que alojó más de 100 millones de sitios web.

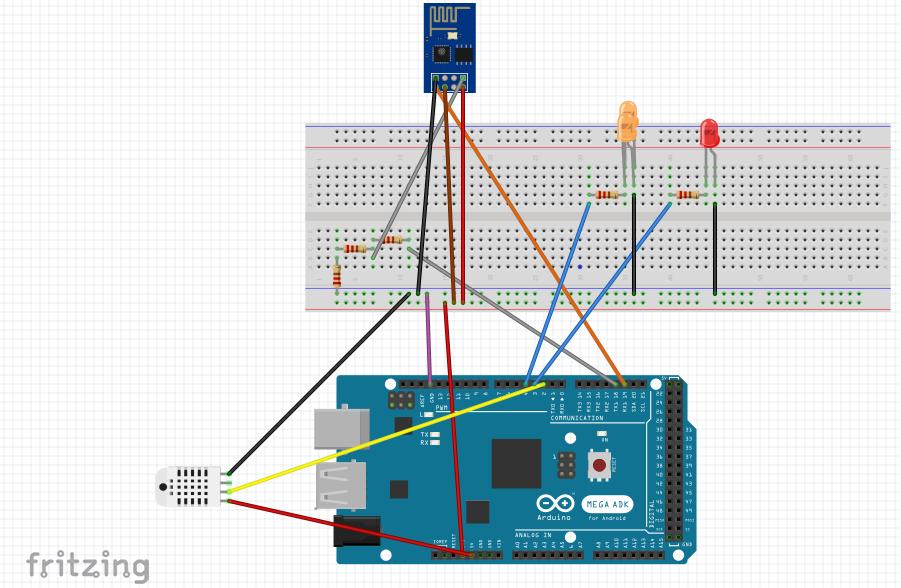
La mayoría de las vulnerabilidades de la seguridad descubiertas y resueltas tan sólo pueden ser aprovechadas por usuarios locales y no remotamente. Sin embargo, algunas se pueden accionar remotamente en ciertas situaciones, o explotar por los usuarios locales malévolos en las disposiciones de recibimiento compartidas que utilizan PHP como módulo de Apache.

**MARCO METODOLÓGICO**

**CONEXIÓN DE COMPONENTES ELECTRÓNICOS**

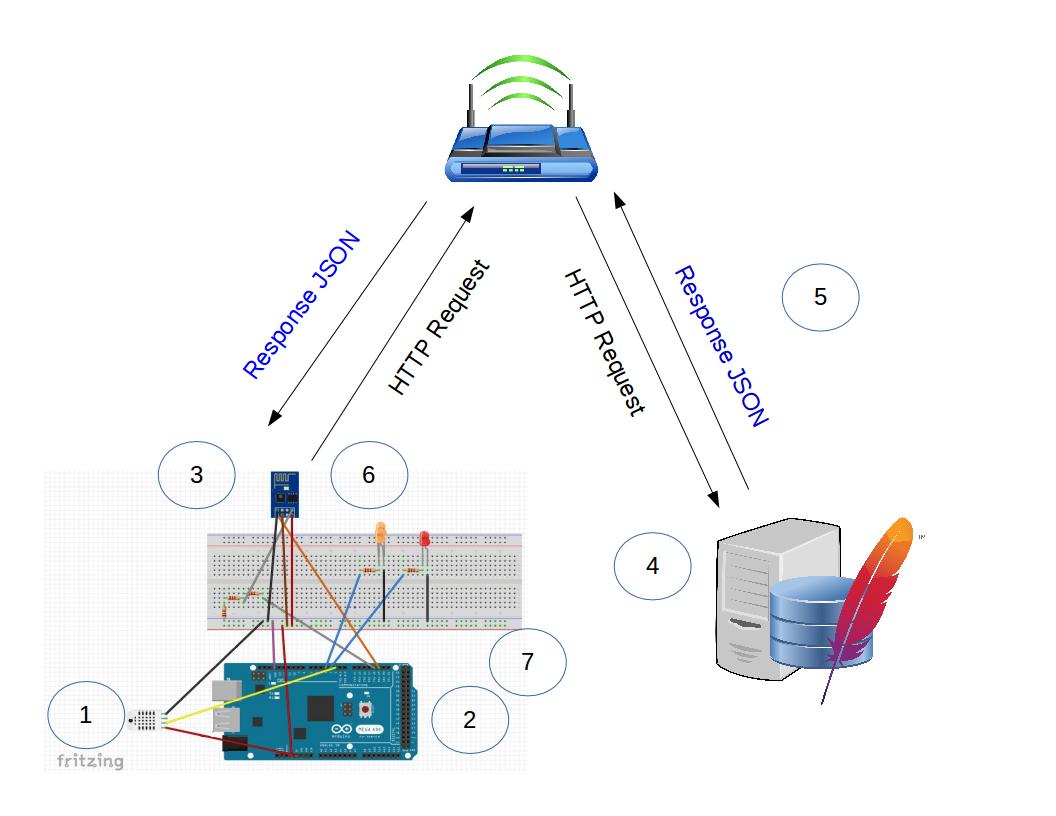
La conexión de componentes electrónicos tiene como dispositivo principal, el Arduino, el cual está conectado al sensor de humedad relativa y temperatura DHT22, al módulo de red inalámbrica ESP8266 y a los actuadores. Para utilizar la comunicación serial por defecto de Arduino, ubicada en los pines 0 y 1 para RX y TX respectivamente, como herramienta para depurar el programa de Arduino, se utilizó el puerto Serial1 para conectar a los pines RX y TX del módulo ESP8266. El puerto Serial1 está disponible en Arduino Mega en los pines 19 para RX y 18 para TX. Para usara el puerto Serial1, o bien Serial2 y Serial3 disponibles en Arduino Mega, para comunicarse con un dispositivo serial TTL externo, se debe conectar el pin TX del dispositivo al pin RX de Mega, el pin RX del dipositivo al TX de Mega y el pin GND a un pin GND del Mega. Como se indicó en el Marco Teórico, el pin CH\_PD y Vcc del ESP8266 deben ir conectados a 3.3V. Por simplicidad se utilizó una salida del Arduino que provee ese voltaje, pero no es lo ideal si estará trabajando por mucho tiempo. Además el pin RX del módulo, fue conectado a la salida lógica 18 del Mega, pero no de manera directa, se conectaron resistencias de 220Ω para regular el voltaje de 5V a 3.3V.

Los demás detalles de la conexión se presentan en el siguiente diagrama.



**MODELO OPERACIONAL**

El modelo operacional del proyecto es presentado y detallado con el siguiente diagrama de despliegue:



1. (Suponiendo que el Arduino está conectado a la red inalámbrica a través del módulo ESP8266 utilizando comandos AT.) El Arduino ordena al sensor DHT22 que le brinde los valores de temperatura y humedad relativa, gracias a funciones que se importan con la biblioteca DHT.h.
2. El Arduino establece conexión TCP por medio del ESP8266 y prepara una solicitud GET por el protocolo HTTP, enviando los valores de temperatura y humedad como parámetros. La solicitud es enviada a través de comandos AT.
3. El módulo ESP8266 envía la petición HTTP al servidor, a través de la red inalámbrica y queda esperando la respuesta.
4. La máquina con software servidor, en este caso se utilizó una laptop con servidor de base de datos MariaDB y servidor Web Apache, recibe la petición HTTP, captura los valores de los parámetros, ejecuta la lógica de negocio y prepara una respuesta empaquetada en formato JSON. Antes de enviar la respuesta, se establece una conexión a la base de datos y se registra el estado del sistema (los valores de las mediciones) y las medidas de mitigación que se emplearán.
5. Se envía la respuesta en formato de notación de objetos de JavaScript o JSON. La respuesta indicará si se debe encender los ventiladores y nebulizadores
6. El módulo ESP8266 recibe la respuesta a la petición y el Arduino la captura en una cadena, usando la conexión Serial1.
7. La cadena de texto con la respuesta contiene todos los elementos de una respuesta HTTP: headers y datos. Lo que interesa recuperar son los datos, los cuales se encuentran en formato JSON, por lo que se implementa una estrategia para localizar el texto JSON. Al localizarse, se convierte la cadena con el texto JSON en un arreglo de caracteres. Para parsear el texto JSON, se utilizaron las funciones de la biblioteca ArduionJson.h. Se crea un buffer estático para almacenar un objeto JSON con dos elementos, se crea un objeto JSON y se le asigna el resultado de parsear el arreglo de caracteres. Si el parseo se ejecutó exitosamente, se capturan los valores para ventiladores y nebulizadores, en variables de tipo entero. Arduino llama a la función actuadores, cuya firma es: void actuadores(int ventiladores, int nebulizadores). La función actuadores activa por quince segundos los ventiladores o los nebulizadores, si el valor de sus variables asociadas es uno, si ambos parámetros reciben valor uno, se activa los nebulizadores por quince segundos e inmediatamente después, se activan los ventiladores por quince segundos. En el modelo creado, los ventiladores son representados por dos ledes y los nebulizadores por una led.

Al finalizar los pasos descritos anteriormente, se regresa al paso uno y se ejecutan en un ciclo continuo mientras exista alimentación eléctrica y por supuesto, la red y el servidor estén disponibles.

**CÓDIGO sketchInvernadero.ino**

#include "DHT.h"

#include <ArduinoJson.h>

String ssid = "CRed";

String password = "a2b4c6d8";

String server = "192.168.43.130"; //IP del server

String uri = "/invernadero/Lectorjson.php"; //uri de php que responde a request de arduino

float temp, hum; //para recibir valores de medicion

String temperatura, humedad;//para el paso de parametros en peticion

#define DHTPIN 2 // Pin donde estÃ¡ conectado el sensor

#define DHTTYPE DHT22 // Sensor DHT22

DHT dht(DHTPIN, DHTTYPE); //variable tipo DHT22

#define pinNebulizadores 3

#define pinVentiladores 4

void setup() {

Serial.begin(115200); //comunicacion arduino-pc

Serial1.begin(115200); /\*comunicacion TTL con ESP8266

Serial1 (RX=19, TX=18)\*/

pinMode(pinNebulizadores, OUTPUT);//para encender nebulizadores

pinMode(pinVentiladores, OUTPUT);//para encender ventiladores

iniciar();

}

void iniciar(){

delay(1000);

String connStr = "AT+CWJAP=\""+ssid+"\",\""+password+"\"";//comando AT para conectarse a una red

Serial1.println(connStr);

delay(5000);

}

void loop() {

/\*Enviar datos desde la consola serial al Serial1 ESp y mostrar

lo enviado por el ESP a nuestra consola.\*/

if(Serial1.available()){

Serial.write(Serial1.read());

}

if(Serial.available()){

Serial1.write(Serial.read());

}

//peticiones

httpget();

delay(30000);

}

void leerDHT(){//lee del DHT la temperatura y humedad y luego construye el objeto arbol en memoria

//DHT

temp = dht.readTemperature();//lectura de temperatura en grados Celcius

hum = dht.readHumidity();//lectura de humedad

/\*temp = 20;

hum = 40;\*/

temperatura = (String)temp;

humedad = (String)hum;

}

void httpget(){

leerDHT();

uri = "/invernadero/Lectorjson.php?temperatura=" + temperatura + "&humedad=" + humedad;

//AT command para iniciar conexion como cliente tipo TCP por el puerto 80

Serial1.println("AT+CIPSTART=\"TCP\",\""+ server +"\",80");

Serial.println("AT+CIPSTART=\"TCP\",\""+ server +"\",80");//echo del comando anterior

if(Serial1.find("OK")){//respuesta en caso de establecer conexion TCP exitosa

Serial.println("Conexion TCP lista.");

}

String getRequest =

"GET " + uri + " HTTP/1.0\r\n" +

"Host: " + server + "\r\n\r\n";

String sendCmd = "AT+CIPSEND=";//comando para enviar datos, seguido se debe escribir la longitud de datos a enviarse, maximo 2048

Serial1.print(sendCmd);

Serial.print(sendCmd);//echo

Serial1.println(getRequest.length());//longitud de los datos que se enviaran, para completar comando AT+CIPSEND

Serial.println(getRequest.length());//echo

delay(1000);

if(Serial1.find(">")){//el comando CIPSEND retorna ">" despues de ejecutarse y esta listo para enviar datos a continuacion

Serial.println("Sending....");

Serial1.print(getRequest);

Serial.print(getRequest);

if(Serial1.find("SEND OK")){//mensaje que se recibe al enviar exitosamente los datos

Serial.println("SEND OK");

String result = "";//contendra el json

boolean httpBody = false; //bandera que indica que parte del response se esta leyendo

while(Serial1.available()){ //leyendo respuesta

String tmp = Serial1.readString();

result = tmp.substring(tmp.indexOf('{'), tmp.indexOf('}')+1);

}

int tam = result.length()+1;

char json[tam];

result.toCharArray(json, tam);

StaticJsonBuffer<JSON\_OBJECT\_SIZE(2)> jsonBuffer;

JsonObject& root = jsonBuffer.parseObject(json);

if(!root.success()){

Serial.println("No se pudo parsear el json.");

}else{

int ventiladores = root["Ventiladores"];

int nebulizadores = root["Nebulizadores"];

Serial.println("Ventiladores: "+ (String)ventiladores + " Nebulizadores : "+(String)nebulizadores);

actuadores(ventiladores, nebulizadores);

}

//cerrar conexion

Serial1.println("AT+CIPCLOSE");//cierra la conexion TCP o UDP

}

}

}

void actuadores(int ventiladores, int nebulizadores){

if(ventiladores == 1 && nebulizadores == 1){

digitalWrite(pinNebulizadores, HIGH);

delay(15000);//enciende nebulizadores por 15 segundos

digitalWrite(pinNebulizadores, LOW);

digitalWrite(pinVentiladores, HIGH);

delay(15000);//enciende ventiladores por 15 segundos

digitalWrite(pinVentiladores, LOW);

}else if(ventiladores == 1 && nebulizadores == 0){

digitalWrite(pinVentiladores, HIGH);

delay(15000);//enciende nebulizadores por 15 segundos

digitalWrite(pinVentiladores, LOW);

delay(15000);//completando los 30 segundos

}else if(ventiladores == 0 && nebulizadores == 1){

digitalWrite(pinNebulizadores, HIGH);

delay(15000);//enciende ventiladores por 15 segundos

digitalWrite(pinNebulizadores, LOW);

delay(15000);//completando los 30 segundos

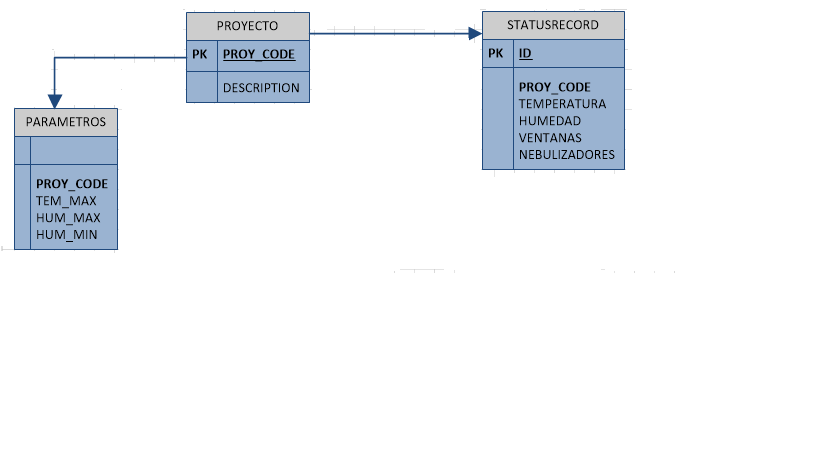
}else{

delay(30000);//equivalente a 30 segundos

}

}

**Diagrama de Entidad – Relación**



**SCRIPT DE LA BASE DE DATOS:**

CREATE TABLE PROYECTO(

PROY\_CODE INT NOT NULL PRIMARY KEY,

DESCRIPTION VARCHAR(30));

CREATE TABLE STATUSRECORD(

ID INT NOT NULL AUTO\_INCREMENT PRIMARY KEY,

PROY\_CODE INT NOT NULL,

TEMPERATURA DECIMAL,

HUMEDAD DECIMAL,

VENTANAS INT,

NEBULIZADORES INT,

FOREIGN KEY(PROY\_CODE) REFERENCES PROYECTO(PROY\_CODE));

CREATE TABLE PARAMETROS(

PROY\_CODE INT NOT NULL,

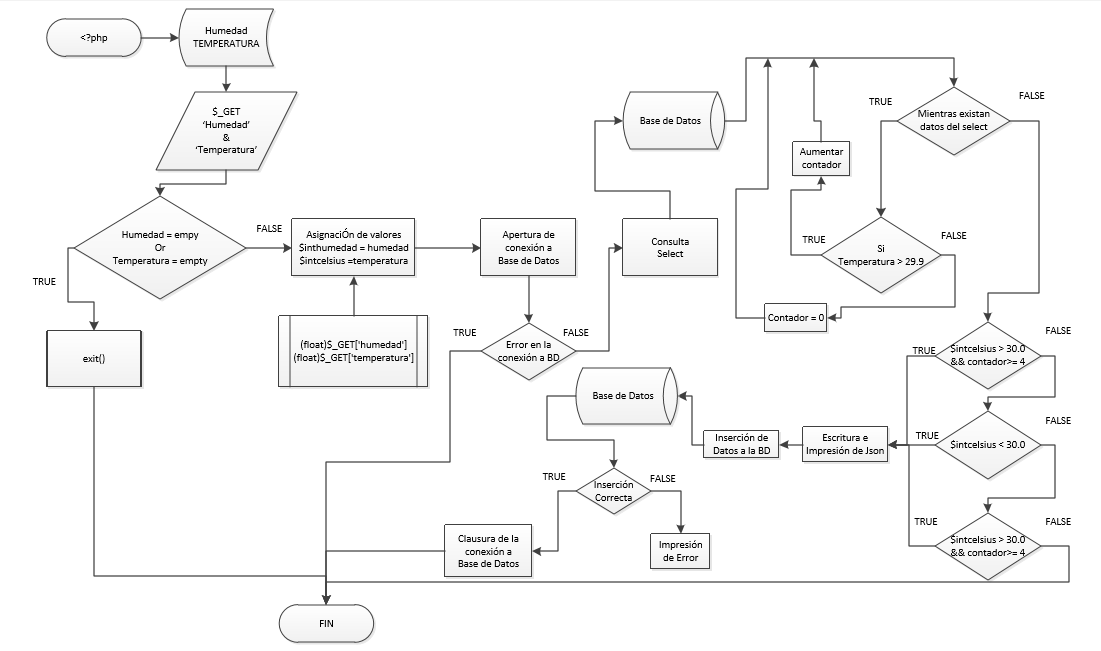
TEM\_MAX DECIMAL,

HUM\_MAX DECIMAL,

HUM\_MIN DECIMAL,

FOREIGN KEY(PROY\_CODE) REFERENCES PROYECTO(PROY\_CODE));

**Diagrama de Flujo**



**Código PHP**

<?php

if(empty($\_GET['humedad']) or empty($\_GET['temperatura'])){

echo "ERROR";

exit();

}

$inthumedad =(float)$\_GET['humedad'];

$intcelsius =(float)$\_GET['temperatura'];

//insertar a Mysql

$dbserver = "127.0.0.1";

$dbuser = "invernadero";

$password = "123456";

$dbname = "sensorial";

$database = new mysqli($dbserver, $dbuser, $password, $dbname);

if($database->connect\_errno) {

die("No se pudo conectar a la base de datos");

}

//Seleccion de datos historicos

$query = "SELECT \* FROM STATUSRECORD ORDER BY ID DESC LIMIT 4";

$result = $database->query($query);

$nu1 = 0;

while ($registro = mysqli\_fetch\_array($result))

{

if($registro['TEMPERATURA'] > 29.9){

$nu1++;

}

else{

$nu1=0;

}

}

if($intcelsius > 30.0 && $nu1 >= 4){

$response = array(

"Ventiladores" => 1,

"Nebulizadores" => 1

);

$json\_response = json\_encode($response);

echo $json\_response;

$sql = "INSERT INTO STATUSRECORD (PROY\_CODE, TEMPERATURA, HUMEDAD, VENTANAS, NEBULIZADORES) VALUES (2016,".$intcelsius.",".$inthumedad.",1, 1)";

if($database->query($sql)== TRUE){

} else {

echo "Error: ".$sql."<br>".$database->error;

}

}

else if($intcelsius < 30.0 ){

$response = array(

"Ventiladores" => 0,

"Nebulizadores" => 0

);

$json\_response = json\_encode($response);

echo $json\_response;

$sql = "INSERT INTO STATUSRECORD (PROY\_CODE, TEMPERATURA, HUMEDAD, VENTANAS, NEBULIZADORES) VALUES (2016,".$intcelsius.",".$inthumedad.",0, 0)";

if($database->query($sql)== TRUE){

} else {

echo "Error: ".$sql."<br>".$database->error;

}

}else if($intcelsius > 30.0 && $nu1 < 4){

$response = array(

"Ventiladores" => 0,

"Nebulizadores" => 0

);

$json\_response = json\_encode($response);

echo $json\_response;

$sql = "INSERT INTO STATUSRECORD (PROY\_CODE, TEMPERATURA, HUMEDAD, VENTANAS, NEBULIZADORES) VALUES (2016,".$intcelsius.",".$inthumedad.",0, 0)";

if($database->query($sql)== TRUE){

} else {

echo "Error: ".$sql."<br>".$database->error;

}

}

$database->close();

**CONCLUSIÓN**

Desarrollamos un modelo del Sistema automatizado para el control de temperatura máxima y humedad relativa en un invernadero para investigación agrícola del Centro Universitario de Oriente. El modelo sirvió para comprobar que podemos prescindir de un computador de placa reducida como Raspberry Pi para la implementación del sistema. Inicialmente se consideró la necesidad de utilizar la Raspberry Pi porque facilitaría la conexión a red y podría alojar al servidor Web y la base de datos. Sin embargo no resulta ideal conectar la Raspberry Pi a sensores porque su sistema operativo es multitarea y no estaría enfocado en esperar la respuesta de la medición y podría perderse algunos resultados. Para evitar este problema se podría conectar la Raspberry Pi a una placa Arduino y que ésta reciba datos de los sensores e interactúe con los actuadores. Sin embargo, utilizar una Raspberry Pi en este proyecto, implica no utilizar adecuadamente los recursos que ésta provee.

Por lo tanto, considerando que gracias a un módulo de conexión a red inalámbrica, el Arduino puede comunicarse con un servidor por el protocolo HTTP, podemos prescindir del uso de una Raspberry Pi. Esto se traduce a una implementación eficiente y económica, ya que el servidor puede alojarse en una computadora de escritorio con recursos limitados para los estándares modernos. Se deberá decidir posteriormente, si se elige una placa Arduino con módulo de red integrado o una placa Arduino básica y un shield para conexión a red.

**REFERENCIAS**

1. Nurdspacenl. (2016). ESP8266. Recuperado el 9 de noviembre de 2016, de <https://nurdspace.nl/ESP8266>
2. Sparkfuncom. (2016). DHT22pdf. Recuperado el 9 de noviembre de 2016, de <https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf>
3. Naylampmechatronicscom. (2016). Tutorial ESP8266 Parte 1. Recuperado el 9 de noviembre de 2016, de <http://www.naylampmechatronics.com/blog/21_Tutorial-ESP8266-Parte-I.html>
4. Arduinocc. (2016). What is Arduino?. Recuperado el 9 de noviembre de 2016, de <https://www.arduino.cc/en/Guide/Introduction>
5. Arduinocc. (2016). Arduino Board Mega2560. Recuperado el 9 de noviembre de 2016, de <https://www.arduino.cc/en/Main/ArduinoBoardMega2560>
6. Arduinocc. (2016). Arduino - Serial. Recuperado el 9 de noviembre de 2016, de <https://www.arduino.cc/en/Reference/Serial>
7. PHP (2016). PHP-Concepto. Recuperado el 9 de noviembre de 2016, de <http://php.net/manual/es/intro-whatis.php>
8. MySQL(2016). MySQL-Concepto. Recuperado el 9 de noviembre de 2016, de <https://es.wikipedia.org/wiki/MySQL>
9. Apache HTTP Server (2016). Apache-Concepto Recuperado el 9 de noviembre de 2016, de <https://es.wikipedia.org/wiki/Servidor_HTTP_Apache>